Minimization and Reduction
Residuals

• The residual of a language $L \subseteq \Sigma^*$ with respect to a word $w \in \Sigma^*$ is the language

$$L^w = \{u \in \Sigma^* \mid wu \in L\}$$

• A language $L' \subseteq \Sigma^*$ is a residual of $L$ if $L' = L^w$ for at least one word $w \in \Sigma^*$

• Observe:

  − $w \in L^u \iff uw \in L$
  − $L^e = L$
  − $(L^w)^v = L^{wv}$
Relation between residuals and states

- Let $A$ be a (finite or infinite) deterministic automaton over an alphabet $\Sigma$.
- The language of a state $q$ of $A$, denoted by $L_A(q)$ or just $L(q)$, is the language recognized by $A$ with $q$ as initial state.
- **Observation 1**: State-languages are residuals.
  - For every state $q$ of $A$: $L(q) = L^w$ for at least one word $w \in \Sigma^*$.
- **Observation 2**: Residuals are state-languages.
  - For every word $w \in \Sigma^*$: $L^w = L(q)$ for at least one state $q$ of $A$. 
Relation between residuals and states
Relation between residuals and states

• Important consequence:

Regular languages have finitely many residuals.

Languages with infinitely many residuals are not regular.
Canonical DA for a language

- Let $L \subseteq \Sigma^*$ be a language (not necessarily regular).

  The canonical DA for $L$ is the tuple

  $$C_L = (Q_L, \Sigma, \delta_L, q_{0L}, F_L)$$

  where

  - $Q_L$ is the set of residuals of $L$, i.e., $Q_L = \{ L^w \mid w \in \Sigma^* \}$
  - $\delta(K, a) = K^a$ for every residual $K \in Q_L$ and $a \in \Sigma$
  - $q_{0L} = L$
  - $F_L = \{ K \in Q_L \mid \epsilon \in K \}$
Canonical DA for a language

• For the language $EE \subseteq \{a, b\}^*$:

$$Q_{EE} =$$

$$q_{0EE} =$$

$$F_{EE} =$$

$$\delta_{EE} =$$
Canonical DA for a language

• For the language $a^*b^* \subseteq \{a, b\}^*$:

$$Q_{a^*b^*} =$$

$$q_0(a^*b^*) =$$

$$F_{a^*b^*} =$$

$$\delta_{a^*b^*} =$$
Canonical DA for a language

• Proposition. $C_L$ recognizes $L$.

• Proof. We prove by induction on $|w| : w \in L$ iff $w \in L(C_L)$

If $|w| = 0$ then $w = \varepsilon$, and we have

$$
\begin{align*}
\varepsilon & \in L & (w = \varepsilon) \\
\iff & L \in F_L & \text{(definition of } F_L) \\
\iff & q_{0L} \in F_L & (q_{0L} = L) \\
\iff & \varepsilon \in L(C_L) & (q_{0L} \text{ is the initial state of } C_L)
\end{align*}
$$

If $|w| > 0$, then $w = aw'$ for some $a \in \Sigma$ and $w' \in \Sigma^*$, and we have

$$
\begin{align*}
aw' & \in L \\
\iff & w' \in L^a & \text{(definition of } L^a) \\
\iff & w' \in L(C_{L^a}) & \text{(induction hypothesis)} \\
\iff & aw' \in L(C_L) & (\delta_L(L, a) = L^a)
\end{align*}
$$
Canonical DA for a language

Theorem. If $L$ is regular, then $C_L$ is the unique minimal DFA up to isomorphism recognizing $L$.

Proof.
1. $C_L$ is a DFA for $L$ with a minimal number of states.
   • $C_L$ has exactly as many states as $L$ has residuals.
   • Every DFA for $L$ has at least as many states as $L$ has residuals.
2. Every minimal DFA for $L$ is isomorphic to $C_L$.

Let $A$ be an arbitrary minimal DFA for $L$. Then:
   • The states of $A$ are in bijection with the residuals of $L$.
   • The transitions of $A$ are completely determined by this bijection: if $q \leftrightarrow L^w$, then $\delta(q, a) \leftrightarrow L^{wa}$
   • The initial state is the state in bijection with $L$.
   • The final states are those in bijection with residuals containing $\epsilon$.
Corollary. A DFA is minimal iff $L(q) \neq L(q')$ for every two distinct states $q$ and $q'$.

Proof.

$(\Rightarrow)$: Let $A$ be a minimal DFA.

Every residual of $L(A)$ is recognized by at least one state of $A$ (holds for every DFA).

Since $A$ is minimal, it has as many states as $C_L$, and so its number of states is equal to the number of residuals of $L(A)$.

Therefore: distinct states of $A$ recognize distinct residuals of $L(A)$.
Corollary. A DFA is minimal iff $L(q) \neq L(q')$ for every two distinct states $q$ and $q'$.

Proof.

($\Leftarrow$): Let $A$ be a DFA such that distinct states recognize distinct languages.

Since every state of $A$ recognizes a residual of $L(A)$, and every residual of $L(A)$ is recognized by some state of $A$ (holds for every DFA), the number of states of $A$ is equal to the number of residuals of $L(A)$.

So $A$ has as many states as $C_L$, and so it is minimal.
Is it minimal?
The Master Automaton

• The master automaton over $\Sigma$ is the tuple $M = (Q_M, \Sigma, \delta_M, F_M)$, where
  
  $Q_M$ is the set of all regular languages over $\Sigma$.
  $\delta_M: Q_M \times \Sigma \rightarrow Q_M$ is given by $\delta_M(L, a) = L^a$.
  $L \in F_M$ iff $\epsilon \in L$.

• The fragment of the Master Automaton containing the states reachable from a state (language) is the canonical DFA for the language.
Plan for the next slides:

1. Computing the language partition
2. Quotienting
3. Thm: The result is the minimal DFA
Computing the language partition
State partitions

- **Block:** set of states.
- **Partition:** set of blocks such that each state belongs to exactly one block.
- Partition $P$ refines partition $P'$ if every block of $P$ is contained in some block of $P'$.
- If $P$ refines $P'$, then we say that $P$ is finer than $P'$, and $P'$ is coarser than $P$.
- **Language partition:** the partition in which two states belong to the same block iff they recognize the same language.
Computing the language partition

• Start with the partition containing (one or) two blocks:
  – Block 1: Final states (accept $\varepsilon$)
  – Block 2: Non-final states (do not accept $\varepsilon$)

• Iteratively split blocks, ensuring that states recognizing the same language always stay in the same block.

• Blocks that contain at least two states recognizing different languages are called unstable.
Finding an **unstable** block

If two states $q_1, q_2$ belong to the same block $B$ but $\delta(q_1, a)$ and $\delta(q_2, a)$ belong to different blocks for some $a \in \Sigma$, then $B$ is **unstable**.
Computing the language partition

**Splitting an unstable block**

We say that \((a, B_1)\) and \((a, B_2)\) are **splitters** of \(B\).

A splitter \((a, B')\) splits \(B\) into two blocks: states \(q\) such that \(\delta(q, a) \in B'\), and the rest.
Splitting an unstable block

We say that \((a, B_1)\) and \((a, B_2)\) are splitters of \(B\). A splitter \((a, B')\) splits \(B\) into two blocks: states \(q\) such that \(\delta(q, a) \in B'\), and the rest.
Correctness

• **Algorithm**: repeatedly pick an unstable block and a splitter, and split the block, until all blocks stable.

• **The algorithm terminates.**

  Every split increases the number of blocks by 1, and the number of blocks is bounded by the number of states.

• **After termination, two states belong to the same block iff they recognize the same language.**

  We show that after termination:

  (1) If two states belong to different blocks, they recognize different languages.

  (2) If two states recognize different languages, they belong to different blocks.
Correctness

(1) If two states $q_1$ and $q_2$ belong to different blocks, they recognize different languages.

By induction on the number $k$ of splittings until $q_1$ and $q_2$ are split (put into different blocks).

• $k = 0$. Then $q_1$ is final and $q_2$ non-final, or vice versa, and we are done.

• $k \to k + 1$. Then there are $q'_1, q'_2$ such that $q_1 \xrightarrow{a} q'_1$, $q_2 \xrightarrow{a} q'_2$, and $q'_1, q'_2$ have been split before $q_1, q_2$ are split.

By induction hypothesis $q'_1$ and $q'_2$ recognize different languages. Since the automaton is a DFA, $q_1$ and $q_2$ also recognize different languages.
Correctness

(2) If two states $q_1$ and $q_2$ recognize different languages, they belong to different blocks.

Let $w$ be a shortest word that belongs to, say, $L(q_1)$ but not to $L(q_2)$. By induction on the length of $w$.

- $|w| = 0$. Then $w = \varepsilon$, $q_1$ is final, and $q_2$ is non-final. So $q_1$ and $q_2$ belong to different blocks from the start.

- $|w| > 0$. Then $w = aw'$ for some $a$, $w'$. Let $q_1' = \delta(q_1, a)$ and $q_2' = \delta(q_2, a)$. Then $L(q_1') \neq L(q_2')$ by the DFA property.

By induction hypothesis $q_1', q_2'$ are put at some point into different blocks.

If at this point $q_1$ and $q_2$ still belong to the same block, then the block becomes unstable and is eventually split.
Quotienting
Definition: The quotient of a NFA $A = (Q, \Sigma, \delta, q_0, F)$ with respect to a partition $P$ is the NFA

$$A/P = (Q_P, \Sigma, \delta_P, q_{0P}, F_P)$$

where

- $Q_P = P$
- $(B, a, B') \in \delta_P$ iff $(q, a, q') \in \delta$ for some $q \in B$ and some $q' \in B'$
- $q_{0P}$ is the block containing $q_0$
- $F_P$ is the set of blocks that contain some state of $F$
Quotient w.r.t. a partition
Quotient w.r.t. a partition
Proposition: The quotient of a DFA with respect to its language partition is (isomorphic to) the canonical DFA.

The proof has two parts:
(1) A DFA and its quotient w.r.t. the language partition recognize the same language.
(2) The quotient is minimal (and therefore the canonical DFA).
Quotient w.r.t. a partition

(1) A DFA and its quotient w.r.t. the language partition recognize the same language.

We prove a more general result (for later use):

**Lemma:** Let $A$ be a NFA, and let $P$ be any partition that refines the language partition $P_l$.

a) For every state $q$: $L_A(q) = L_{A/P}(B)$, where $B$ is the block containing $q$.

b) If $A$ is a DFA and $P = P_l$, then $A/P$ is also a DFA.
Quotient w.r.t. a partition

a) For every state \( q \) of \( A \): \( L_A(q) = L_{A/P}(B) \), where \( B \) is the block containing \( q \).

We prove that for every word \( w \in \Sigma \):

\[
  w \in L_A(q) \iff w \in L_{A/P}(B).
\]

By induction on \( |w| \).

- \( |w| = 0 \). Then \( w = \epsilon \) and

  \[
  \epsilon \in L_A(q) \quad \text{iff} \quad q \in F \\
  \text{iff} \quad B \subseteq F \\
  \text{iff} \quad B \in F_P \\
  \text{iff} \quad \epsilon \in L_{A/P}(B)
  \]

  (because \( P \) refines \( P_\ell \))
Quotient w.r.t. a partition

a) For every state $q$ of $A$: $L_A(q) = L_{A/P}(B)$, where $B$ is the block containing $q$.

- $|w| > 0$. Then $w = aw'$. There is $q \xrightarrow{a} q'$ in $A$ such that $w' \in L_A(q')$. There is $B \xrightarrow{a} B'$ in $A/P$ such that $q' \in B'$.

We have:

$aw' \in L_A(q)$ iff $w' \in L_A(q')$ (Def. of $q$)
  iff $w' \in L_{A/P}(B')$ (induction hyp.)
  iff $aw' \in L_{A/P}(B)$ ($B \xrightarrow{a} B'$)
Quotient w.r.t. a partition

b) If $A$ is a DFA and $P = P_l$, then $A/P$ is also a DFA.

We show: If $B \xrightarrow{a} B_1$ and $B \xrightarrow{a} B_2$, then $B_1 = B_2$.

- There are $q, q' \in B$, $q_1 \in B_1$, $q_2 \in B_2$ such that $q \xrightarrow{a} q_1$ and $q' \xrightarrow{a} q_2$.
- Since $P = P_l$, $q$ and $q'$ recognize the same language.
- Since $A$ is a DFA, $q_1$ and $q_2$ recognize the same language.
- Since $P = P_l$, $B_1 = B_2$. 
The quotient of a DFA $A$ w.r.t. the language partition is the canonical DFA.

- By 1.b, the quotient is a DFA.
- By 1.a, applied to the initial state, $A/P_\ell$ recognizes the same language as $A$.
- Since the quotient is w.r.t. the language partition, different blocks of the quotient recognize different languages. So $A/P$ is minimal.
Hopcroft´s algorithm

• The algorithm for the computation of the language partition is nondeterministic: It does not specify which unstable block to split next.
• Hopcroft´s algorithm is a refinement that carefully chooses the split order, and achieves a complexity of $O(mn \log n)$ for a DFA with $n$ states over an $m$-letter alphabet.
• The algorithm maintains a workset of possible splitters.
Hopcroft´s algorithm

• The algorithm maintains a workset of candidate splitters \((a, B)\).

• When a candidate \((a, B)\) is taken from the workset, it is applied to all current blocks.

• **Observation 1**: After applying \((a, B)\) to all blocks it never brings anything to apply it again

  ⇒ it is safe to ensure that candidates removed from the workset are never added to the workset again.

• **Observation 2**: If \(B\) is split into \(B_0\) and \(B_1\), then splitting w.r.t. any two of \((a, B), (a, B_0), (a, B_1)\) produces the same result as splitting with respect to all three.
Hopcroft's algorithm

Hopcroft(A)

Input: DFA $A = (Q, \Sigma, \delta, q_0, F)$

Output: The language partition $P_\ell$.

1. if $F = \emptyset$ or $Q \setminus F = \emptyset$ then return $\{Q\}$
2. else $P \leftarrow \{F, Q \setminus F\}$
3. $\mathcal{W} \leftarrow \{(a, \min\{F, Q \setminus F\}) \mid a \in \Sigma\}$
4. while $\mathcal{W} \neq \emptyset$ do
5. pick $(a, B')$ from $\mathcal{W}$
6. for all $B \in P$ split by $(a, B')$ do
7. replace $B$ by $B_0$ and $B_1$ in $P$
8. for all $b \in \Sigma$ do
9. if $(b, B) \in \mathcal{W}$ then replace $(b, B)$ by $(b, B_0)$ and $(b, B_1)$ in $\mathcal{W}$
10. else add $(b, \min\{B_0, B_1\})$ to $\mathcal{W}$
11. return $P$
Reducing NFAs
Minimal NFAs are not unique
Finding minimal NFAs is hard

**Theorem:** The following problem is PSPACE-complete: Given an NFA $A$ and a number $k$, decide if there is another NFA $B$ equivalent to $A$ and having at most $k$ states.

**Proof idea:** We will show later that the following problem is PSPACE complete: given an NFA $A$ over alphabet $\Sigma$, decide whether $L(A) = \Sigma^*$. The problem above can be reduced to this one. This shows PSPACE-hardness.
Reducing NFAs

We wish to use the same idea as before:

• Compute a suitable partition \( P \) of the states of the NFA.
• Quotient the NFA with respect to this partition.

Requirements on \( P \):

• \( L(A) = L(A/P) \)
• Efficiently computable
Partitions suitable for reduction

• Recall: For every NFA $A$ and partition $P$ that refines the language partition: $L(A) = L(A/P)$.

• So any such partition is good for reduction.

• A partition refines the language partition iff states in the same block recognize the same language (states in different blocks may not recognize different languages, though!).

• (Observe: Such partitions refine the partition $\{F, Q \setminus F\}$.)
Computing a suitable partition

- **Idea**: use the same algorithm as for DFA, but with new notions of **unstable** block and block splitting.
- **We must guarantee:**
  
  after termination, states of a block recognize the same language
  
  or, equivalently
  
  after termination, states recognizing different languages belong to different blocks
If $L(q_1) \neq L(q_2)$ then either
- one of $q_1, q_2$ is final and the other non-final, or
- one of $q_1, q_2$, say $q_1$, has a transition $a_{\rightarrow} q_1'$ such that every $a$-transition $a_{\rightarrow} q_2'$ satisfies: $L(q_1') \neq L(q_2')$. 

The key observation
Unstable blocks

A block $B$ is **unstable** if there are states $q_1, q_2 \in B$, a block $B'$ and $a \in \Sigma$ such that

$$\delta(q_1, a) \cap B' \neq \emptyset \quad \text{and} \quad \delta(q_2, a) \cap B' = \emptyset$$

We say that $(a, B')$ splits $B$. 

[Diagram of a state transition graph showing two blocks $B$ and $B'$, with states $q_1$ and $q_2$, and transitions labeled $a$.]
Splitting blocks

Splitting an unstable block

We say that \((a, B')\) is a splitter of \(B\).

A splitter \((a, B')\) splits \(B\) into two blocks: states \(q\) such that \(\delta(q, a) \cap B' \neq \emptyset\), and the rest.
Splitting blocks

Splitting an unstable block

We say that \((a, B')\) is a splitter of \(B\).
A splitter \((a, B')\) splits \(B\) into two blocks: states \(q\) such that \(\delta(q, a) \cap B' \neq \emptyset\), and the rest.
An example
An example
The algorithm not always computes the language partition

States 2 and 3 recognize the same language: $c(d + e)$
However, the algorithm puts them into different blocks.